

## REMARKS

Claims 1-32 are pending in the present application. Reconsideration of the claims is respectfully requested.

### **I. 35 U.S.C. § 102, Anticipation**

The Office Action rejects claims 1-32 under 35 U.S.C. § 102 as being anticipated by *Kolawa et al.* (U.S. Patent No. 5,784,553); hereinafter referred to as “*Kolawa*”. This rejection is respectfully traversed.

The Office Action states:

As per claim 1, 4 and 17, *Kolawa et al* teach the path is one of the path is shown in column 6 line 39-42 (“Each program is broken down into a series of code blocks comprising one or more program statements occurring along a single path of execution”), plurality of paths is shown in column 2 line 42-44 (“every branch at least once and optionally for generating as many paths as desired in the total path coverage set”), for the rest of the limitations see the rejection of the previous office action.

(Office Action dated October 3, 2000, pg. 2).

Amended claim 1 reads as follows:

1. A process in a data processing system executing a routine having a plurality of paths, wherein the routine has includes a plurality of first type instructions and wherein the data processing system executes second type instructions, the process comprising:
  - identifying a path being executed, wherein the path is one of the plurality of paths in the routine and wherein a plurality of first type instructions are associated with the path; and
  - translating the first type instructions for the path being executed, wherein first type instructions are translated into second type instructions for execution by the data processing system, wherein first type instructions for unexecuted paths within the routine remain untranslated.

Amended claim 4 reads as follows:

4. A process in a data processing system for executing a method having a plurality of paths, wherein the data processing system executes native machine code, the process comprising:
  - identifying a path being executed, wherein the path is one of the

plurality of paths in the within the method and wherein a plurality of bytecodes are associated with the path; and

compiling bytecodes for the path being executed, wherein the bytecodes are compiled into native machine code executed by the data processing system, wherein bytecodes for unexecuted paths within the method remain uncompiled.

Amended claim 11 reads as follows:

11. A process in a data processing system for executing a method having a plurality of paths in which each path with in the plurality of paths contains a number of bytecodes, the method comprising:  
identifying the method that is to be executed; and  
compiling the bytecodes into instructions for execution by the data processing system for each path within the plurality of paths as each path is executed.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that the Office Action has not presented a case of anticipation because the reference, *Kolawa*, does not teach the claimed subject matter recited in independent claims 1, 4, 11, 17, 24 and 30, as discussed hereafter. Independent claims 17, 24 and 30 include similar features as independent claims 1, 4 and 11.

Initially, the Office Action contains selected portions of *Kolawa* without regard to their actual arrangement in order to manufacture an anticipation rejection of the present invention. For example, in alleging anticipation of the feature as recited in claim 1 “identifying a path being executed, wherein the path is one of the plurality of paths in the routine”, the Office Action cites two unassociated sections of *Kolawa*. First, the Office Actions cites, for the feature of “the path is one of the path”:

Each program is broken down into a series of code blocks comprising one or more program statements occurring along a single path of execution.

(*Kolawa*, col. 6, lines 39-42).

Then the Office Action cites for the feature of “plurality of paths”:

... every statement at least once and for taking substantially every branch at least once and optionally for generating as many paths as desired in the total path coverage set.

(*Kolawa*, col. 6, lines 42-44).

When taken together and analyzed the two referenced sections of *Kolawa* actually do not teach identifying a path being executed, wherein the path is one of the plurality of paths in the routine or identifying a path, wherein the path is one of a plurality of paths within the method as recited in claims 1 and 4 of the present invention. The first section referenced by the Office Action refers to breaking down a program statement into a series of code blocks thereby facilitating storage of information extracted from the original computer program (*Kolawa*, col. 6, lines 37-39). The second section referenced by the Office Action refers to finding an input for causing a program statement to execute (*Kolawa*, col. 2, lines 38-40). Neither of these two referenced sections of *Kolawa*, either separately or combined teach identifying a path being executed, wherein the path is one of the plurality of paths in the routine or within the method as recited in claims 1 and 4. Both referenced sections are concerned with a program statement after identification of the program statement. This situation is apparent when *Kolawa* is viewed in its entirety rather than in the piece meal fashion as presented by the Office Action. Therefore, *Kolawa* does not teach the step of identifying a path being executed, wherein the path is one of the plurality of paths or within the method.

For the translating step in claim 1, the Office Action references the following section of *Kolawa*:

Yet another embodiment of the present invention is a computer-implemented method of finding inputs that will generate runtime errors in a computer program written in the JAVA programming language, the computer program being represented by JAVA bytecodes after being compiled by a JAVA

compiler. The method includes the steps of reading the JAVA bytecodes; obtaining an input value for the computer program; symbolically executing an instruction of the computer program represented in bytecodes; ...

(*Kolawa*, col. 3, lines 34-42.)

The above reproduced section of *Kolawa* does not teach the translating step. This section, at most, teaches that a JAVA bytecode is read, obtaining an input value and executing an instruction of the computer program represented in bytecodes by using the input value. This section associates a bytecode instruction with an input value to simulate operation of the computer program instruction. No translating takes place. The bytecode is not converted from one form to another. The input value is not converted from one form to another. The input value is simply associated with the bytecode computer instruction to perform the program instruction. Therefore, *Kolawa* does not teach the step of translating the first type instructions for the path being executed as recited in claim 1 of the present invention.

In addition, the Office Action references the following for the proposition that *Kolawa* teaches the feature of unexecuted paths remaining untranslated:

The original source code 11 comprises all types of files used to express an uncompiled, that is, non-object code, computer program, including definitional and declarative files.

(*Kolawa*, col. 5, lines 53-55).

However, *Kolawa* further explains:

For example, in the C programming language, header files are declarative files since they frequently declare but do not define program variable, structures and functions. Source code rules, however, are definitional files since they typically contain definitions of program variables, structures and functions.

(*Kolawa*, col. 5, lines 55-61).

Therefore, in context, what the referenced section of *Kolawa* states is that some files cannot be compiled because of their very nature, such as containing only information about the program. This is not the same as the feature in the present invention as recited in claim 1 wherein first type instructions for unexecuted paths within the routine remain untranslated. The files referenced in *Kolawa* do not contain instructions and, hence,

because of their very nature remain uncompiled whether they are being executed or not. In contrast, the present invention as recited in claim 1, where instructions in unexecuted paths remain untranslated because the instructions remain unexecuted. Therefore, *Kolawa* does not teach the step of wherein first type instructions for unexecuted paths within the routine remain untranslated as recited in claim 1 of the present invention.

Further, *Kolawa* does not provide any teaching or suggestion of translating the instructions from one type of instruction to another type of instruction for a path being executed as recited in claim 1. The presently claimed invention translates instructions from one type into another as being executed. Based on the teaching of *Kolawa*, the translation of instructions occur through a compiling process, which does not involve execution of the program. Thus, when *Kolawa* is viewed as a whole for what it teaches, rather than in a piece meal fashion, each and every feature of the presently claimed invention is not shown in *Kolawa*, as arranged in claims 1 and 4.

In addition, the section cited by the Office Action in regard to compiling the bytecodes as recited in claim 11:

FIG. 23 is a flow diagram of the steps of handling JAVA bytecode instrumentation. At step 492, the JAVA Parser 230 reads the .class files 210 generated by the JAVA Compiler 208 and the .java files 200 of the original source code....

(*Kolawa*, col. 26, lines 44-47).

JAVA bytecode instrumentation in *Kolawa* does not compile the bytecode as recited in claim 11 of the present invention. JAVA bytecode instrumentation is used for profiling code by adding bytecodes that keep track of what portions of the code get executed (*Kolawa*, col. 26, lines 22-24). A JAVA virtual machine can be used to execute the instrumented bytecodes in order to gather extra information about the JAVA program under test (*Kolawa*, col. 26, lines 41-43). In contrast, JAVA bytecode instructions are generated that are non-specific to a particular computer architecture (Specification, pg. 4, lines 8-9). The section referenced by the Office Action allegedly teaching compiling the bytecodes merely reads a file. There is no teaching of compiling of bytecodes in this referenced section of *Kolawa*. Furthermore, the Office Action has not pointed out in *Kolawa* or any other prior art where compiling the bytecodes into instructions is taught as

recited in claim 11. In addition, the Office Action has not pointed out in *Kolawa* or any other prior art where compiling the bytecodes into instructions for each path as the path is executed as recited in claim 11.

Further, independent claims 17, 24 and 30 and the claims dependent upon claims 1, 4, 1, 17, 24 and 30 are patentable over this cited reference for the same reasons. Further, the dependent claims include other additional combinations of features not taught or suggested by the reference. For example, claims 3, 6, and 13 include an additional step of storing the compiled instructions in an execution order. It does not necessarily follow that translated instructions will be stored in an order of execution in *Kolawa* or any other prior art. In contrast, the instructions may be stored in the same order in which they are found in the method or routine in claims 3, 6 and 13. Consequently, it is respectfully urged that the rejection of claims 1-32 under 35 U.S.C. § 102 have been overcome.

Further, the presently claimed invention in claims 1-32 are not obvious in view of *Kolawa*. No teaching, suggestion, or incentive is present for one of ordinary skill in the art to make the necessary changes to *Kolawa* to reach the presently claimed invention. When *Kolawa* is viewed as a whole, *Kolawa* teaches compiling or translating a program in its entirety. The program statement, the portion of which the Office Action relies on for the alleged identifying of the path, is concerned with a program statement after identification of the program statement. Therefore, *Kolawa* teaches away from the presently claimed invention since only after the identification of the program statement are code blocks generated or inputs found. As discussed above, a teaching or interpretation of identifying a path being executed while it is being executed as recited in claims 1 and 4 is not present in *Kolawa*.

Further, one of ordinary skill in the art would not be motivated to make these changes when the problems addressed by *Kolawa* and the present invention are considered by one of ordinary skill in the art. The present invention recognizes that problems exist with just in time compiling in which entire methods are compiled regardless of whether they are used (Specification, pg. 5, lines 1-3). The presently claimed invention is concerned with the problem of trying to reduce use of data processing system resources during execution of methods. In contrast, *Kolawa* is

concerned with difficulties in testing software and in particular creating a good test suite (*Kolawa*, col. 1, lines 28-46). Thus, one of ordinary skill in the art would not be motivated to look *Kolawa*, much less modify *Kolawa* with respect to the presently claimed invention.

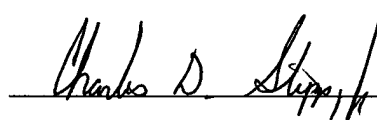
Therefore, the presently claimed invention may be reached only through an improper use of hindsight with the benefit of Applicants' invention as a template to piece together and interpret teachings from *Kolawa*. No teaching, suggestion, or incentive is present for the interpretations or modifications needed to reach the presently claimed invention.

## II. Conclusion

Applicant respectfully submits that all of the claims are directed to allowable subject matter and that the application is in condition for allowance. Should the Examiner believe anything further is necessary to place this application in even better condition for allowance, the Examiner is requested to contact Applicant's representative at the telephone number listed below.

DATE: December 15, 2000

Respectfully submitted,



Charles D. Stepps, Jr  
Reg. No. 45,880  
Carstens, Yee & Cahoon, LLP  
P.O. Box 802334  
Dallas, TX 75380  
(972) 367-2001  
ATTORNEY FOR APPLICANT